VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

GENEROVÁNÍ UŽIVATELSKÉHO ROZHRANÍ PRO WORKFLOW SYSTÉM

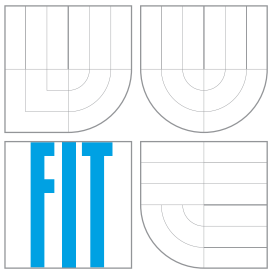
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

KATEŘINA ŠÍMOVÁ

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

GENEROVÁNÍ UŽIVATELSKÉHO ROZHRANÍ PRO WORKFLOW SYSTÉM

GENERATING OF USER INTERFACE FOR WORKFLOW SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

KATEŘINA ŠÍMOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MILAN POSPÍŠIL

BRNO 2013

Abstrakt

Tato bakalářská práce se zabývá automatickým generováním uživatelského rozhraní z definovaných metadat, zejména generováním uživatelských formulářů. Dále řeší úpravy metadat a čtení a ukládání uživatelských dat ve formulářích. V práce se obecně pojednává o pojmu workflow a workflow systémech, pro které implementovaný generátor určený. Práce rozebírá možnosti preprezentace workflow dat a vhodné technologie k implementování zadaného problému.

Abstract

Bachelor's thesis concerns with automatic generation of user interfaces from defined metadata, more particularly generation of user forms. It also enables editing of metadata and reading and saving of user's data in forms. In theoretical part there are discussed terms workflow and workflow systems for which the implemented generator is dedicated. Thesis also discusses options of presentation of workflow data and appropriate technologies that can be used to implementation of given assignment.

Klíčová slova

Workflow, Grafické uživatelské rozhraní, .NET, C#, XML, Windows Forms, Modelování dat

Keywords

Workflow, Graphic user interface, .NET, C#, XML, Windows Forms, Data modeling

Citace

Kateřina Šímová: Generování uživatelského rozhraní
pro Workflow systém, bakalářská práce, Brno, FIT VUT v Brně, 2013

Generování uživatelského rozhraní pro Workflow systém

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Milana Pospíšila

.....
Kateřina Šímová
14. května 2013

Poděkování

Děkuji svému vedoucímu Ing. Milanu Poslíšilovi za odborné vedení přii vypracování této práce a za podněty, které mi poskytl.

Obsah

Kapitola 1

Úvod

Začátkem celého řetězce událostí, které vedou k myšlence zavedení workflow systému do firmy je její konkurence schopnost. Motorem celého procesu je snaha produkovat lepší produkty za méně peněz. Většina firem, které přemýšlejí o nasazení workflow systému již prošla většinou kroků jako modernizací výrobního vybavení, restrukturovala své lidkské zdroje a mohla provés například kompletní BPR (Business Process Reengeneering) na jehož základě optimalizovala firemní procesy. Nyní přichází krok automatizace firemních procesů, tedy využití workflow systému.

Zakoupení a instalování workflow systému však zdaleka nené řešením. Nejdůležitější součástí je správné uvedení do provozu, správné modelování firemních procesů nebo třeba takypropojení s dalšími aplikacemi či propojení se stávajícími databázovými záznamy firmy.

Protože větčina workflow systémů je velmi robustních a univerzálních je potřeba je pro koncového uživatee kustomizovat. Takovéto přispůsobení svým potřebám může uživatel provést do jisté míry sám, namodelováním firemních procesů do workflow systému. Takový uživatel však bude omezen předdefinovanými vlastnostmi systému. Druhou možností je kustomizovat uživateli produkt ještě před jeho instalací a to malými úpravami, většinou už ve zdrojových kódech systému. Navržená a implementovaná aplikace - generátor uživatelského rozhraní workflow systému - je určena právě k prototypování změn prováděných při kustomizaci systému. Je určena pro společné použití vývojáře (či prodejce) systému a jeho budoucího uživatele. Aplikace by měla umožnit jak vizualizaci dat workflow systému tak ukázat jakým způsobem je možno s těmito daty manipulovat. Generátor může samozřejmně zůstat součástí i hotového systému. [1]

Kapitola 2

Workflow

Workflow je pojem, který se používá k označení všeho, co se týká automatizace řízení procesů firmy, od návrhů rozhodovací logiky a jejích digaramů, přes komplexní řešení firemních informačních systémů, až po informace putující skrze firemní procesy. Často jsou pojmem workflow označovány konkrétní implementace nástrojů využívajících logiku automatizovaných firemních či výrobních procesů. Přesněji můžeme uvést, že workflow je: "Tok informací v podnikovém procesu a jejich automatizované řízení."[1]

Podnikový proces (Bussines Process) v sobě zahrnuje všechny druhy procesů, firem libovolného zaměření, ať už se jedná o procesy výrobní, či ty, které se týkají lidských zdrojů, nebo v nejnáročnější variantě procesy rozhodovací. Podle definice instituce Workflow Management Coalition je podnikový proces: "Množina propojených procedur nebo aktivit, které společně realizují firemní cíl a firemní politiku v kontextu organizační struktury definující funkční role a vztahy."[3]

Často se pojem workflow nesprávně užívá jako přídavné jméno při označení aplikací, které jen do jisté míry pracují s procesy a pravidly nad nimi a vytvářejí tak jakousi workflow logiku. Nejsou však plnohodnotnými workflow systémy. Tyto aplikace můžou zdánlivě splňovat níže uvedenou definici pojmu workflow, ale jako systém nemají veškeré součásti a náležitosti workflow systému tak, jak je definován v druhém odstavci kapitoly 2.1. Toto přídavné jméno by tedy mělo označovat schopnost daného informačního systému využívat logiku workflow.

Pojem workflow byl definován v roce 1996 institucí Workflow Management Coalition v terminologickém slovníku jako: "Automatizace celého nebo části podnikového procesu, během kterého jsou dokumenty, informace nebo úkoly předávány od jednoho účastníka procesu k druhému podle sady procedurálních pravidel tak, aby se dosáhlo nebo přispělo k plnění celkových podnikových cílů."[3]

Někdy bývá nesprávně zaměňován pojem workflow s BPR (Business Process Reengineering) [12], tedy pozorování, studování a následné inovování a optimalizování firemních procesů. Při BPR se často využívá informačních technologií, kvůli kterým je právě zaměňováno s workflow (v obecném pojetí smyslu tohoto pojmu). Na teoretické úrovni je však možno provést BPR i bez pomoc informačních technologií. Avšak BPR a workflow jsou spolu do jisté míry propojeny a to takovým způsoben, že zavedení workflow systému může být důsledkem BPR.

2.1 Workflow systémy

Pod pojmem workflow systém budeme v následujícím textu zjednodušeně chápat informační systém, který dokáže automatizovaně řídit firemní procesy a poskytuje další služby s tím spojené. Jako je využívání databází, práce s externími dokumenty a aplikacemi či poskytování možnosti grafického návrhu map workflow procesů.

Podle instituce Workflow Management Coalition (dále jen WfMC) je workflow systém, jinak také *Systém řízení workflow* v textu Terminology & Glossary označen anglickým termínem *Workflow Management System*, definován jako: "Systém, který definuje vytváří a řídí provádění workflow za pomoci užití softwaru běžícím na jednom nebo více workflow jádrech, který je schopný interpretovat definici procesu, interagovat s účastníky workflow a tam, kde je to nutné, vyvolá spuštění IT nástroje a aplikace."[3]

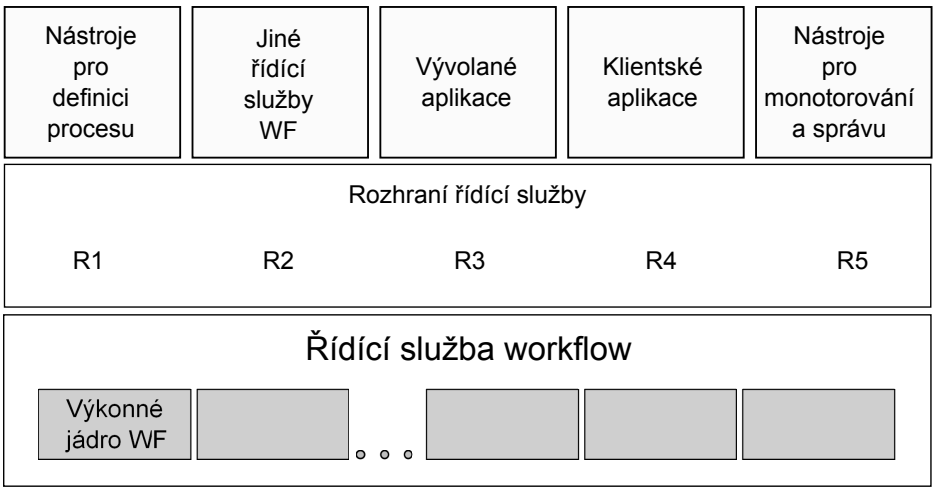
Jednoduše znázorněné součásti workflow systému vidíme i na obrázku 2.1.Podrobněji schematicky popsaný workflow systém můžeme najít v [3, str. 39, obr. 5]. Kopii obrázku naleznete i v příloze B.2.

Ve workflow systému je potřeba exaktně definovat obecně známé procesy firmy v podobě vhodné pro automatické zpracování. Pro definici procesu existuje standart: *metamodel definice procesu* vytvořený institucí WfMC, který rozděluje definici procesu na jednotlivé objekty a zobrazuje vazby mezi nimi. Viz. [1, str. 40 obr. II.8.].

2.1.1 Rozhraní workflow systému

Pojem rozhraní workflow systému má stejný význam jako obecný pojem rozhraní pro libovolný informační systém. Jedná se o množinu možností, kterými můžou do systému přicházet a ze systému vycházet data. V případě workflow systému se jedná většinou o vstupy jiných aplikací (systémů), vstupy souborů (dokumentů), generování e-mailů a uživatelská rozhraní. Často se setkáváme s rozhraními pro práci s informacemi z relační databáze, napojení workflow akcí přímo na poštovní klienty a především s uživatelskými formuláři.

Rozlišit bychom měli rozhraní celého systému a rozhraní řídící služby workflow, která zapouzdřuje jeno nebo více workflow jader. Podle [1] se můžeme bavit o pěti rozhraních řídící služby workflow.



Obrázek 2.1: Rozhraní čídící služby workflow

2.2 Workflow data

Workflow systémy pracují s třemi typy dat. Jedním druhem jsou řídící data workflow (anglickým termínem: Workflow control data), druhým jsou specifická workflow data nazývána také jako Věcná data workflow nebo anglickým pojmem jako Workflow relevant data a třetím typem jsou data využívaná aplikacemi, tedy Aplikační data. Každé z těchto dat můžeme zařadit také pomocí

Řídící data

"Workflow control data obsahují informace o vnitřních stavech vztažené k různým procesům a instance aktivit, které jsou spouštěny a mohou také obsahovat informace nutné k obnovení/restartování workflow enginů v případě selhání nějaké aktivity."[9]

Věcná data

Každá věcná data workflow musí obsahovat údaje o svém názvu, typu a cestě, po které budou putovat. Pokud takováto data nemají explicitně uvedený typ, řídící služba workflow ani jádro workflow s nimi nemůžou pracovat. Můžou je však předat dál pro zpracování v některé specifické aplikaci.

Pro představu takováto data můžou být například informace, které budou nějakým automatizovaným nástrojem získána například z příchozího e-mailu a předána skrze rozhodovací logiku dalším účastníkům workflow, kterými může být například účetní program nebo konkrétní výrobní stroj. V takovém případě však může být více než vhodné, aby byla data zkontrolována či upravena člověkem. Právě na zobrazení těchto dynamicky se měnících dat v průběhu vykonávání dat workflow v různých místech workflow je zaměřen implementovaný generátor, který využívá i přístup k datům z relační databáze.

Aplikační data

Tyto data jsou těmi daty, se kterými pracují koncové aplikace a uživatelé těchto aplikací. Tyto data nejsou dosažitelná řídící službou workflow, jsou to data aplikací za rozhraním řídící služby workflow.

2.2.1 Propojení s relačními daty

S daty z databází mohou pracovat všechny typy workflow dat. Pro implementaci generátoru uživatelských rozhraních bude důležité propojení dat z relační databáze a metadat věcných dat workflow systému. Data z databáze mohou sloužit pro ukládání neměnících se výčtů, které budou často využívány nebo zobrazovány. Druhou možností je poskytování uživateli výběrů z dat v databázi pro vyplnění formulářů, například v podobě vložení výsledku SQL dotazu do pole nabídky uživatelského vstupu typu ComboBox, nebo jako poskytovaná nápověda při vyplňování vstupů, například v podobě takzvaného našeptávání.

2.2.2 Datové typy

Jak je zmíněno výše, data putující skrze workflow systém mají vždy jasně daný svůj typ. Pojem datový typ definujme jako: "množinu hodnot spolu s operacemi, které je možné nad těmito hodnotami provádět"[10]. Použité typy jsou omezeny pouze konkrétním implementačním řešením a samotný význam workflow je nijak neovlivňuje. V teoretické rovině se tedy můžeme bavit o všech dostupných datových typech ve smyslu jejich obecného významu v IT. Budeme rozlišovat datové typy atomické a datové typy složené. Častěji než typy atomické se využívají složené datové typy, jako jsou především kolekce a struktury, dále se pak setkáváme také s dokumenty. Přehled datových typů ve vztahu k existujících jádrům workflow vytvořili autoři [8]. Jejich přehled je znázorněn v tabulce B.1.

Kapitola 3

Modelování dat

Modelování dat chápejme jako možnosti reprezentování skutečnosti v prostředí počítače, tedy přímo v programu nebo v externích souborech. Při modelování dat pro nějaký informační systém musí v prvé řadě dojít k redukci dat popisujících modelovanou skutečnost pouze na data, která jsou pro navrhovaný systém relevantní a pak musí být správně zvolen způsob, jakým budou data reprezentována. Pro reprezentaci dat v programu se bude jednat například o datový typ, nebo v případě objektového modelování správně navržená třída pro ukládaní data to pak budou databáze nebo souborové systémy. Od vytvořeného datového modelu se pak budou odvíjet i možnosti jejich vizualizace v uživatelských rozhraních.

3.1 Data a metadata

Pro potřeby modelování dat je potřeba vymezit pojem data a metadata, která spolu úzce souvisí a odlišit je od sebe. Pod pojmem data chápeme takovou reprezentaci skutečnosti, která je schopna přenosu, uchování, interpretace či zpracování. Data většinou nemají sémantiku. Ta data, která sémantiku mají, jsou pak nazývána informacemi. Metadata je pojem označující soubor všech použitých datových typů. Metadata určují, jak budou data, ke kterým se vztahují, vypadat. Metadat může existovat vždy více úrovní. Úroveň metadat se označuje mocninou. Označení se zapisuje například takto: meta^{2}data. Nultá mocnina metadat označuje samotná data a metadata úrovně n+1 určují obecně zapsanou podobu (vlastnosti) metadat nižší mocniny. Zjednodušeně pak data reprezentují konkrétní hodnoty a jejich metadata první mocniny určují struktury, podle kterých se řídí veškerá možná práce s daty.

Dobrým příkladem pro užití dat a metadat je rozdíl mezi dokumenty XML a XSD (čili XML Schema Definition). Zatímco v části dokumentu XML budou párové značky označovat začátek a konec jedné struktury a v nich budou mezi dalšími párovými značkami již konkrétní informace o datech této struktury, v XSD dokumentu bude určeno, jak se zmíněné párové značky jmenují a jaké další párové značky se mohou použít mezi nimi. U těchto značek poslední úrovně (v terminologii stromů "listů") může být pak určeno, jakého typu je informace, která je v XML dokumentu nesena mezi danými párovými značkami. [7]

Následují příklady kusů kódu XML souboru a souboru XML schéma definující XML nad ním.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<person rc=9006135436>
<Given name>Franta</Given name>
```

```
<Surname>Alfonzů</Surnamei>
<Date of birth>13.06.1990</Date of birth>
</person>
```

První ukázka kódu znázorňuje úryvek z klasického XML dokumentu. Obsahuje několik základních informací o položce typu "Osoba", která je v systému vedena pod svým rodným číslem.

V následující ukázce je pak úryvek z XSD souboru pro předchozí data. V tomto kódu jde vidět, jak definuje prvky pro data o osobách a atribut záznamu, kterým je rodné číslo osob. Všechny definice zde zahrnují název a datový typ.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="person">
<xs:complexType>
<xs:sequence>
<xs:element name="Given name" type="xs:string"></xs:element>
<xs:element name="Surname" type="xs:string"></xs:element>
<xs:element name="Date of birth" type="xs:date"></xs:element>
</xs:sequence>
<xs:attribute name="rc" type="xs:int"></xs:attribute>
</xs:complexType>
</xs:element>
</xs:schema>
```

3.2 Struktury a kolekce

Nejčastěji využívaným způsobem reprezentace dat ve vrstvě věcných dat workflow systému jsou struktury a kolekce. Většinou se jedná o kolekce struktur obdobně jako obecně v informačních systémech. V případě objektově orientovaného návrhu budou struktury i kolekce reprezentovány vhodně připravenými třídami, kde aby v případě definice dat nedošlo k rekurzivní definici budou využity třídy nesoucí metadata pro tyto struktury a kolekce. V případě ukládaných dat se jedná o data v podobě stromu zapsaná v jazyce XML nebo ve formátu JSON (JavaScript Object Notation), který je schopný pojmout a na řetězec převést libovolně složitou strukturu či kolekci dat, také nezávisle na tom jaké obsahuje datové typy. Oba tyto formáty je možné použít pro přenos dat například po síti. Formátu JASON se velmi využívá při přenosech informací při webových aplikacích [4].

Jak je uvedeno výše jedná se o složené datové typy, tedy skládající se z n-tic prvků jakéhokoli datového typu. Pod pojmem struktura pak budeme chápat n-tici prvků obecně různých typů. Struktura bývá téže nazývána jako katalog. Jednotlivé prvky struktury jsou nazývány vlastnostmi (anglicky property).

Kolekci pak budeme chápat jako n-tici prvků stejného typu. O kolekci můžeme také mluvit jako o multimnožině. Důležitou vlastností kolekcí je jejich možnost mít přiřazeno jedno nebo více uspořádání jejích prvků. Abychom mohli kolekci uspořádat, musí být určen klíč, podle kterého se bude toto uspořádání řídit. Tyto klíče mohou být obecně složené i víceúrovňové. [10]

Kapitola 4

Grafiké uživatelské rozhraní

Tvorba grafického uživatelského rozhraní je jedním ze dvou stěžejních bodů této práce. Veškeré GUI je velmi úzce spojeno s reprezentací dat, která mají být vizualizovaná či čtena z uživatelského vstupu. Důležitým faktem v této práci je, že větší část uživatelského rozhraní je zobrazovýna dynamicky, tedy podle dat a metadat, která nejsou při kompilaci programu známa.

4.1 Vizualizace a čtení dat

Většina grafických frameworků poskytuje obdobné možnosti pro zobrazování dat uživatelům, stejně tak i komponenty, do kterých můžou uživatelé data vepisovat. V následujícím textu budeme uvádět příklady konkrétních komponent z jazyka, ve kterém byla implementována aplikace, která je předmětem této zprávy, tedy C# a grafického frameworku Windows Forms. Více o možných grafických frameworcích pro jazyk C# v kapitole 6.1.3 .

Zmíněné komponenty do jisté míry vyžadují přesně určený typ zobrazovaných dat, je však poměrně obvyklé přetypování hodnot, které mají být uživateli zobrazeny. Takové přetypování může být vyžadováno například u komponenty "label", která při operaci přiřazení do své vlastnosti "Text"(což je text, který se zobrazí na panelu, na který byla komponenta vložena) vyžaduje datový typ řetězec. Komponenta "label"je však například vhodná i k jednoduchému zobrazení dat typu integer. Proměnnou typu integer však není možno do vlastnosti "Text"přiřadit bez předchozího přetypování. Takovéto přetypování je plně v režii programátora.

Obdobný příklad bychom mohli uvést v případě komponenty "TextBox", do které může uživatel vepsat libovolnou hodnotu, která je reprezentována jako typ řetězec. V takovém případě je vhodné provést validaci vstupů, tedy zkontrolovat, zda do polí očekávající celé číslo, byla opravdu vložena data, která můžou být reprezentována jako proměnná typu integer. Následně musí dojít k přetypování, které může být doprovázeno detekováním a rozdělováním vstupu (anglicky parse) podle šablony hodící se k načítanému typu.

4.1.1 Formuláře

Formuláře jsou velmi často používanou součástí grafických uživatelských rozhraní pro aplikace jak webové, tak desktopové. Formulář je takový soubor grafických komponent, které uživateli připravují jakousi šablonu pro data, která má předat programu. Formulář se skládá z textů, které uživatele informují o hodnotách, které mají být vepsány, samotných kompo-

nent, které přijímají vepisované hodnoty, a většinou potvrzovacího tlačítka, které spustí událost, která započne načítání dat do programu.

Výhodou formulářů v elektronické podobě, oproti papírovým, je možnost využití grafických komponent, které mohou uživateli pomoci s výběrem nebo upřesněním hodnoty, kterou je potřeba zadat a samozřejmě možnost validace dat a okamžité žádání jejich opravy. Správně zvolené a nastavené komponenty ve formuláři mohou velmi napomoci jednoduchosti jeho zpracování a kvalitě získaných dat. Pokud by jednotlivé komponenty a zobrazované vysvětlující názvy přesto nebyly pro uživatele úplně jednoznačné, je vhodné doplnit tyto části formuláře o "ToolTipy", tedy doplňující informace zobrazující se po najetí kurzorem na některou z komponent nebo popisků.

Mezi komponenty pomáhající uživateli s vyplňováním hodnot můžou patřit následující: Skupiny "Radiobuttonů"a "Checkboxy"při jejichž správném použití můžeme uživatele jasně omezit, kolik možností z výběru může zvolit. Jako možnost výběru z několika položek můžeme také použít komponentu "Combobox", která výrazně sníží prostor potřebný k zobrazení všech možností této položky ve formuláři oproti "Radiobuttonům". U "Comboboxů"je mnoho možností na nastavení, pro formuláře a omezení uživatele z pohledu zadání nevalidních informací je nápomocná možnost, nedovolit vepisovat do pole "Comboboxu"vlastní hodnoty. Další užitečnou komponentou může být komponenta pro vkládání časových údajů "DateTimePicker", která zajistí stejný formát časového údaje a může uživateli usnadnit výběr data zobrazením kalendáře a nápovědou aktuálního data. Dále mohou být využity či doprogramovány další uživatelsky příjemné doplňky jako například šipky zvyšující a snižující hodnotu čísla zadávaného do komponenty "TextBox"nebo pro zvýšení přehlednosti použit některý z kontainerů (např.: "GroupBox") a mnohé další.

Pomocí nastavení komponent formuláře je možno také ošetřit takzvaná povinná pole, tedy taková pole, bez jejichž vyplnění nelze formulář odeslat ke zpracování. Takováto povinná pole by měla být řádně graficky označena aby uživatel nebyl nemile překvapen výstražnými hlášeními o jejich povinnosti až v momentě vyplnění formuláře a pokusu o jeho odeslání. Možností pro vyznačení takovýchto polí je v současné době ve zvyku více. Mezi nejčastější patří označení popisu pole symbolem hvězdičky nebo barevné odlišení pole. [2]

4.1.2 Vizualizace kolekcí a struktur

Nejpodstatnější částí vizualizace workflow dat pro uživatele je vizualizace kolekcí a struktur. Pro přehledné a jednoduché zobrazení celé struktury či kolekce může být využito speciálních komponent, nebo můžou být zobrazovány pomocí komponent jednoduchých, které budou nějakým způsobem sdružovány do skupin pomocí komponent, ze skupiny kontainerů. Vhodnou komponentou pro grafické sdružení při zobrazování dat ve struktuře může být například komponenta "Groupbox", která poskytuje grafické ohraničení své oblasti a možnost zobrazení pojmenování této skupiny (vlastnost "Text").

Zmíněné komponenty použitelné speciálně pro zobrazování struktur a kolekcí mohou být například tabulky nebo zobrazení stromů. Tabulky jsou vhodné zejména pro zobrazení kolekcí atomických typů nebo kolekcí strukturu obsahující pouze atomické položky. Naopak pro samotné struktury, jejichž prvky jsou neatomické jsou vhodná zejména stromová zobrazení, která výborně vystihují hierarchii struktury. Pro zobrazení hierarchických dat můžeme využít také komponent jako je "LinkLabel", tedy textu s hypertextovým odkazem, který pomocí události navazující na kliknutí na tuto komponentu zobrazí další obsah - například nižší úroveň stromu nebo podrobnosti jednotlivého záznamu kolekce. Prvky "LinkLabel"můžou být využity i při zobrazování kolekcí neatomických hodnot případně kolekcí

struktur neatomických hodnot pro zobrazení zanořeného obsahu.

4.2 Generování uživatelského rozhraní na základě metadat

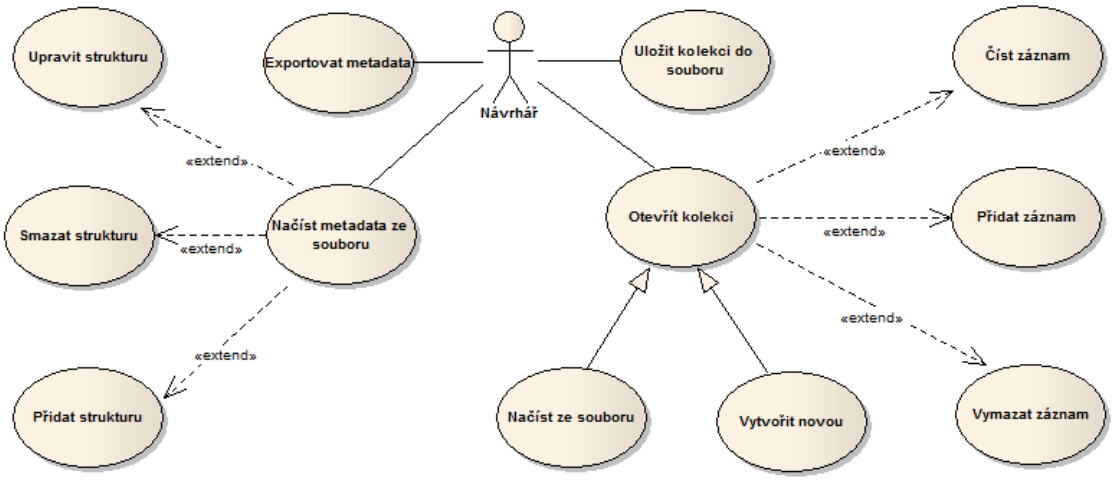
Při generování uživatelského rozhraní na základě metadat se jedná o situaci, kdy nemáme určená konkrétní data, která se mají zobrazovat, ale máme pouze šablonu, podle které se může zobrazovat větší množství dat. Využití takovéhoto přístupu může být vhodné například pro uživatelské formuláře definované metadaty, v jejichž polích pak můžou být zobrazována konkrétní data nebo můžou být ponechány prázdné a tak připravené pro přijetí dat zadaných uživatelem.

Kapitola 5

Analýza a návrh

Implementovaná aplikace, Generátor uživatelského rozhraní pro Workflow systém, byla navržena jako modul pro workflow systém, sloužící k vizualizaci věcných dat workflow a prototypování univerzálního workflow systému při jeho kustomizaci a přípravě nasazení do konkrétní firmy. Aplikace by měla kromě vizualizace dat poskytovat uživateli možnost pracovat především s jejich metadaty a ukazovat aplikované změny v metadatech na konkrétních kolekcích dat pro větší názornost. Dále by měl mít uživatel možnost pro lepší představu práce s výslednou podobou workflow systému, také pracovat přímo se samotnými daty v podobě kolekcí. Všechna data i metadata by měla být uložitelná do XML souboru pro pozdější využití.

Základní úkony, které by měly být dostupné uživateli tohoto modulu, jsou znázorněny v následujícím diagramu 5.1.

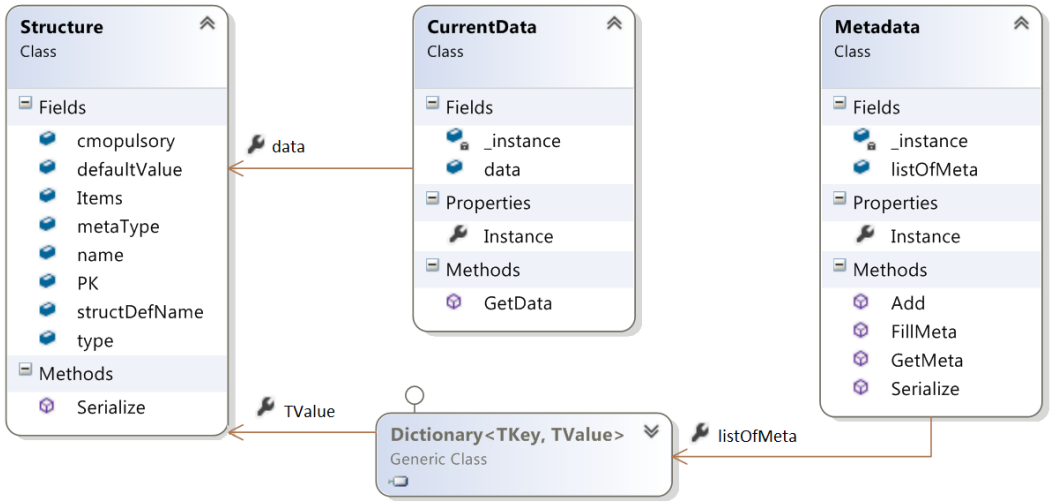


Obrázek 5.1: Use Case diagram navrhované aplikace

5.1 Práce s daty a metadaty

Aplikaci můžeme rozdělit na dvě části podle funkcionality. Jednu zacházející s daty (na diagramu 5.1 vpravo) a druhou zacházející s metadaty (na diagramu 5.1 vlevo). Obě za-

hrnují podobné operace o jejichž rozdílech bude dále psáno v kapitole 6. Jsou to operace importování a exportování dat či metadat, přidání, odebrání a úprava jednotlivých dílčích součástí dat a metadat a samozřejmě jejich procházení a zobrazování. Pro odlišnosti ve způsobu reprezentace dat a metadat, a tak i operací nad nimi jsou pro tyto dvě základní datové jednotky aplikace modelovány dvě třídy (na obrázku 5.2 uvedeny jako CurrentData a Metadata). Obě tyto třídy využívají třídu Structure, která znázorňuje hiearachičnost zpracovávaných dat a metadat. Třída Structure znázorňuje také jednotlivé položky dat a metadat.



Obrázek 5.2: Diagram reprezentujícíc třídy pro data a metadata

5.1.1 Data

Data zpracovávaná v generátoru jsou specifikací určeny jako kolekce záznamů. Například kolekce záznamů typu "Osoba"se bude chovat podobně jako databázový záznam (tabulka Osoba), jako data řízená metadaty má však větší potenciál dynamičnosti. Takovýmto datům můžeme v běhu aplikace přidávat možnosti - rozšiřovat či ubírat informace, které nesou, úpravami jejich metadat.

Třída CurrentData modeluje zpracovávanou kolekci, tato kolekce může být v jedné instanci programu pouze jedna. Proto je tato třída modelována podle návrhového vzoru Singleton.

5.1.2 Metadata

Metadata mají uvádět jakého typu a v jakých strukturách - jak hierarchicky uspořádány - mají být data, na která budou aplikovány. Zobrazení metadat pro uživatele má pak být reprezentováno jako prázdné šablony (formuláře) pro data. Pro aplikaci je také navrženo speciální zobrazení metadata, mimo hlavní okno aplikace, tak, aby co nejlépe vystihlo hierarchii metadat a pomohlo uživateli se lépe zorientovat při jejich editaci či vytváření dat. Třída Metadata modeluje metadata, podle kterých se řídí zpracovávaná data. Ke každým

datům mohou v jednom čase náležet pouze jedna metadata, a proto je tato třída modelována podle návrhového vzoru Singleton. U této třídy, která je definicí dat je důležité aby nedošlo k rekurzivní definici.

5.1.3 Struktura

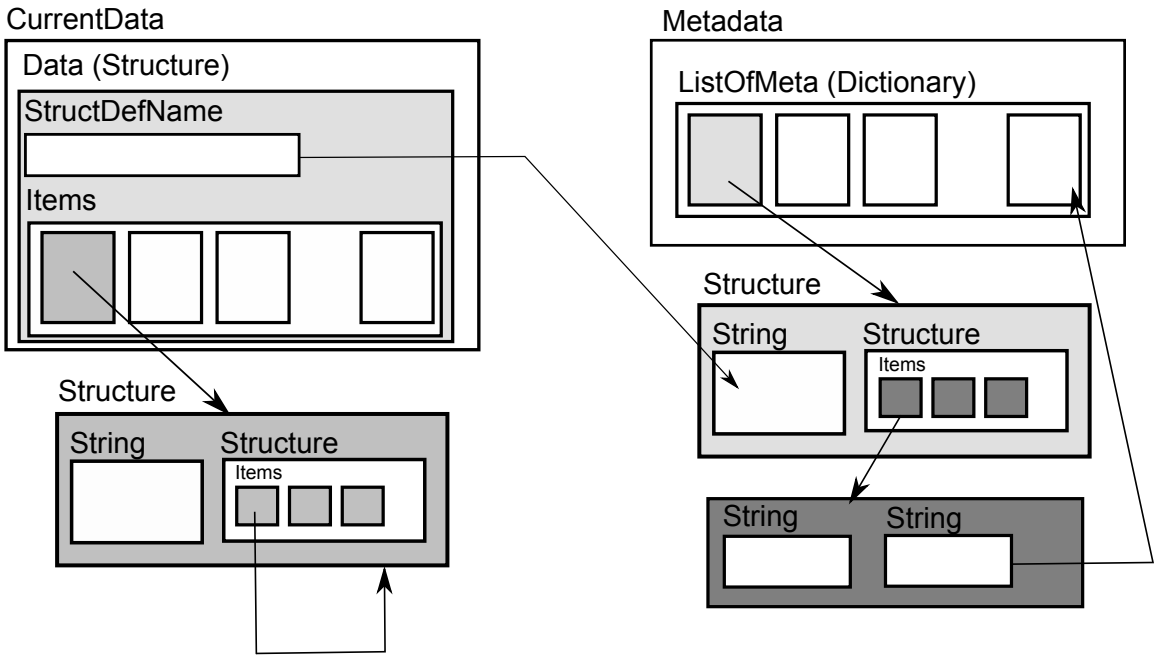
Třída Structure znázorňuje jeden prvek dat nebo metadat. Její atribut "Items", který je seznamem instancí třídy Structure, pak dovoluje jednotlivé prvky řadit hierarchicky pod sebe a tak vystavět logický strom. Tento princip funguje stejně pro data i metadata, ale kvůli zachování nerekurzivní definice dat bude v implementaci pro metadata seznam obsahovat pouze klíče do struktury metadat. Třídu struktura můžeme využít na modelování tří skutečností.

Zaprvé pro modelování stromu dat. Zde jde o čisté instance třídy Structure, které v atributu Items (seznamu) obsahují další instance této třídy. Tento strom je modelován třídou CurrentData. Pokud se jedná o koncový prvek stromu, nese tento prvek vlastnosti jako je jeho jméno, výchozí hodnota a název struktury, podle které se řídí.
V případě uzlu tohoto stromu reprezentovaných dat, který je strukturou podle definice z 3.2 nese prvek kromě zmíněných vlastností listu ještě seznam prvků, které obsahuje (položka Items) a pokud je tento prvek kolekcí podle 3.2 nese ještě hodnotu svého primárního klíče.

V druhém případě modeluje třída Structure konkrétní případ metadat, která budou aplikována na jednotlivá data, přesněji na jednotlivé položky - listy a uzly stromu změněného v předcházejícím odstavci. Zde se jedná o jednu instanci třídy Structure, která v atributu Items nese klíče do kolekce dostupných metadat. Strom dat je s těmito metadaty propojen ve svém kořeni, kde podle atributu structDeffName (tedy klíče definujícího metadata pro tuto strukturu), tyto metadata ve svém seznamu Items nesou klíče do kolekce metadat (následující odstavec) a nedochází tak k rekurzivní definici dat.

V třetím případě modeluje třída Structure dostupná metadata. Přesněji kolekce instancí třídy Structure modeluje kolekci všech dostupných metadat pro jednotlivé struktury i podstruktury dat. Zmíněná kolekce je reprezentována třídou Metadata. V tomto případě je seznam Items využíván jako jednoúrovňový seznam. V tomto seznamu se hledá pomocí klíčů zmíněných výše.
Pro modelování metadat jsou důležité informace jméno, datový typ, jestli je u této informace ve formulářích vyžadováno její vyplnění (položka compulsory) a samozřejmě položka items jako podstruktury těchto metadat.

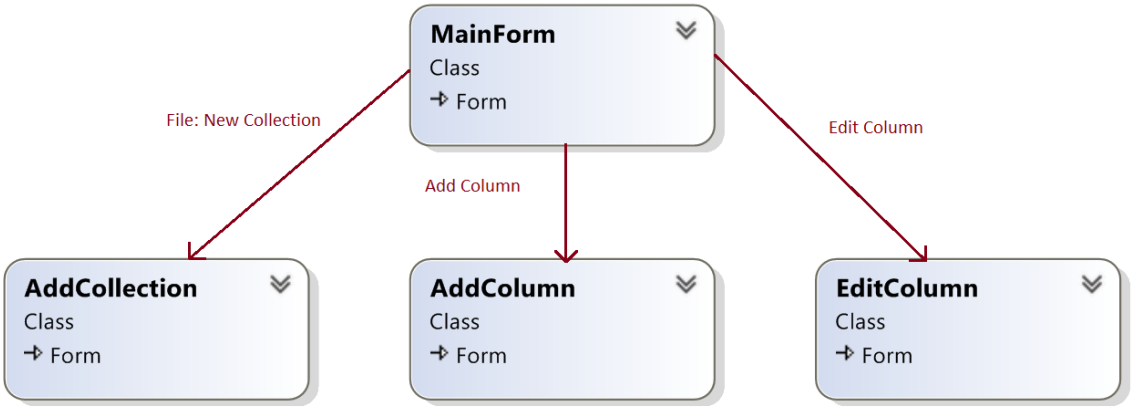
Rozdíl mezi modelováním dat a metadat pomocí třídy Structure je znázorněn na následujícím obrázku 5.3.



Obrázek 5.3: Vztah dat a metadat a jejich využití třídy Structure

5.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní bude rozděleno do 3 částí - tří oken aplikace. První částí je Hlavní formulář, ve kterém se uživateli budou zobrazovat generované komponenty a data. Druhou částí budou formuláře manipulující s daty a třetí částí formuláře manipulující s metadaty. Konkrétně se jedná o formuláře znázorněné na obrázku 5.4 s přechody - tlačítky, které je vyvolávají. Pro potřeby generování uživatelských rozhraní, konkrétně pro zobrazování vstupních polí pro čísla v uživatelských formulářích, byla také navržena třída NumberBox 5.2.1.



Obrázek 5.4: Třídy (fomuláře) grafického uživatelského rozhraní

5.2.1 Třída NumberBox

Třída numberbox je třídou, která bude sloužit pro vizualizaci formulářů, konkrétně pro pole uživatelského vstupu, který má být dále využíván jako číslo. Tato třída dědí za třídy TextBox a omezuje uživatelské vstupy do jejích instancí pouze na čísla, celá i reálná.

5.2.2 Hlavní okno aplikace

Hlavní formulář (v diagramu jako MainForm) je tím, který je uživateli zobrazen při spuštění aplikace. Tento formulář nese klasické menu horní lišty s položkami Soubor (File), Zobrazení (View) a Struktury (Structures). Menu File umožňuje importování a exportování dat i metadat a obsahuje možnost "New"pro vytváření nových kolekcí. Menu View umožňuje změnit způsob zobrazování dat v střední části okna (která je na obrázku vynechána). Základním zobrazením je zobrazení hierarchie dat pomocí jejich zařazení do grafických prvků typu GroupBox a grafické odsazování od okraje okna. Druhou možností je zobrazení pomocí textu a hypertextových odkazů, které otevírají zanořené struktury do nových oken aplikace. Tato nová ona aplikace je již bez menu horní lišty a slouží pouze k zobrazení podstruktur.

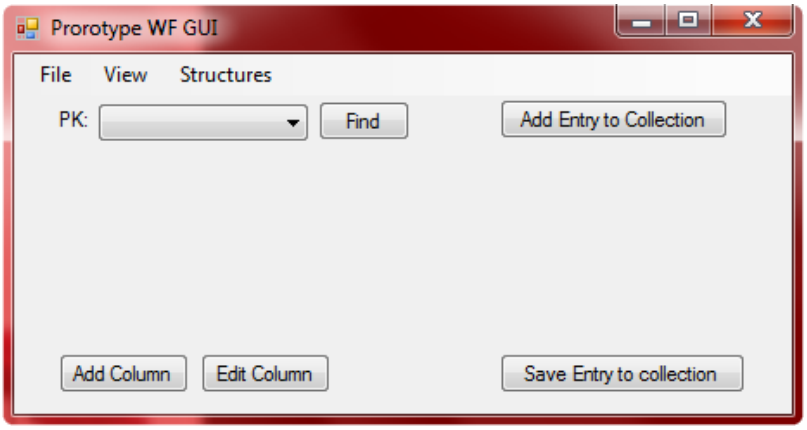
Volba "New"v položce menu horní lišty "File"nabízí možnost "New Collection", která slouží uživateli k vytvoření nové kolekce na základě některých z načtených metadat. Stisknutí tohoto tlačítka vede k otevření nového okna pro vytváření kolekcí. Toto okno je v

diagramu 5.4 označeno jako: AddCollection. Druhou možností po položkou "New"je "Metadata". Tato volba vymaže aktuálně načtená metadata a otevře dialog "Add Item", kterým uživatel vytvoří metadata nová. Použití tohoto formuláře je blíže popsáno v kapitole 5.2.3.

Pod menu horní lišty se nachází řádek pro práci s kolekcemi. Poté, co je kolekce načtena, může uživatel hledat v jejích prvcích pomocí selekčního prvku "PK"podle primárního klíče. Za tímto polem následuje tlačítko pro přidaní nového prvku (záznamu) do otevřené kolekce. Toto tlačítko přepíše stávající obsah středu hlavního okna prázdným formulářem podle načtených metadat. Po vyplnění formuláře (více v kapitole 6) může uživatel data uložit pomocí tlačítka "Save Entry to Collection". Toto tlačítko je navrženo na umístění do pravého dolního rohu z důvodu větší uživatelské přívětivosti a pohodlnosti situace, kdy uživatel projde všechna pole formuláře a dostane se tak až na spodní okraj okna a může ve stejném prostoru vkládací činnost i ukončit uložením svých nových dat.

Poslední součástí hlavního okna jsou dvě tlačítka pro práci s metadaty navrženy do levého dolního rohu. Tato tlačítka by se mohla nacházet i pod některou položkou horní lišty aplikace, ale v situaci, kdy z podstaty této aplikace budou používána velmi často je uživatelsky přívětivější, aby byla stále dostupná.Tato dvě tlačítka vedou k otevření nových oken aplikace a to okna pro přidání nového záznamu do načtených metadat v případě tlačítka "Add Column"a okna pro úpravu a mazání načtených metadat v případě tlačítka "Edit Column". Tato okna jsou v diagramu 5.4 označena jako AddColumn a EditColumn.

Prostřední část formuláře, která je na obrázku 5.5 nahrazena bílou mezerou je místem, kde se zobrazují veškeré formuláře ve všech zobrazovacích variantách. V případě zobrazování do nového okna se zde zobrazuje pouze kořenový uzel stromu struktur.



Obrázek 5.5: Výřez z hlavího okna aplikace

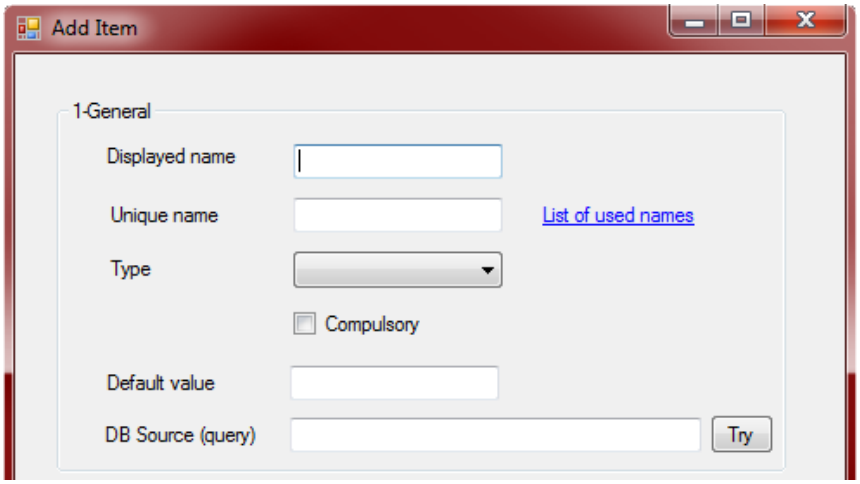
5.2.3 Vkládání struktur do metadat

Zpracovávání editování metadat je navrženo do dvou oddělených formulářů. Jeden pro přidávání struktur k metadatům a druhý pro úpravu stávajících metadat, který zahrnuje i jejich mazání. Všechny tyto úkony jsou vykonávány nad metadaty připojenými ke zpracovávané kolekci, nikoli nad celým souborem dostupných metadat.

Přidávání struktur k metadatům slouží formulář "AddColumn", který provede uživatele čtyřmi kroky, ve kterých vytvoří strukturu a umístí ji na zvolené místo v hierarchii stávajících struktur. Bylo zvažováno použití průvodce (tzv. Wizardu), ve kterém by uži-

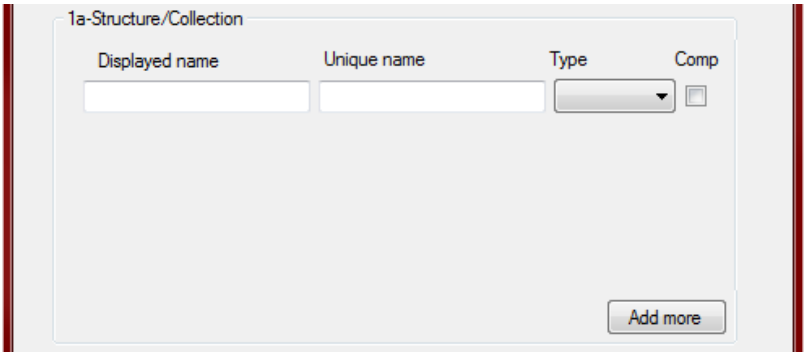
vatel vyplňoval informace a po každém kroku se přesunul na další pomocí tlačítka další (”Next”), nakonec však především z důvodu větší přehlednosti a po přihlédnutí k faktu, že kroků při přidávání struktury není příliš mnoho, byla zvolena varianta, kdy jsou všechny kroky zobrazeny na jednom formuláři a jejich logické celky jsou od sebe odděleny grafickým znázorněním komponenty GroupBox.

V prvním kroku vyplní uživatel obecné informace o přidávané struktuře. Její jméno, které bude zobrazováno v uživatelském rozhraní, unikátní jméno struktury, které bude tuto strukturu identifikovat v kolekci struktur (metadata). Pro ulehčení volby unikátního jména může uživatel nahlédnout do hierarchie struktur, která je dostupná po kliknutí na hypertextový odkaz vedle tohoto pole. Dále může zvolit výchozí hodnotu, která bude použita pokud tato položka atomického typu v datech řídících se podle upravované struktury, nebude mít uloženou jinou hodnotu a u položky atomického typu uživatel zvolí jestli je toto pole povinné. Atomičnost přidávané struktury se rozpoznává podle zvolené varianty v poli Typ. Pokud uživatel vybere v tomto poli neatomický typ (strukturu nebo kolekci) budou pole pro zadání povinnosti a výchozí hodnoty a pole pro připojení informací z databáze nedostupné. Pokud uživatel vybral typ přidávaná struktury ”Enum”bude mu poskytnuta možnost vepsat do posledního pole SQL dotaz, který vybere s připojené databáze informace, které se budou zobrazovat jako možnosti tohoto nově přidávaného výčtu.



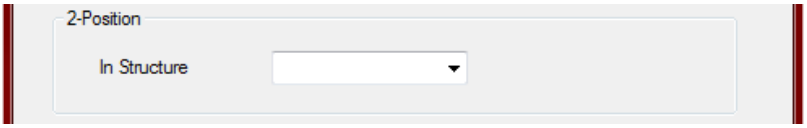
Obrázek 5.6: Výřez z okna pro přídávání struktur: první krok

Druhý krok (označený v aplikaci jako 1a) je uživateli zpřístupněn pouze, pokud zvolil jako typ vkládané struktury neatomický typ. V tomto poli může zadávat položky do seznamu podpoložek přidávané struktury. Tyto položky se zadávají pouze stručným způsobem, aby výsledný formulář nebyl příliš rozsáhlý. Tento způsob je vhodný pro rychlé zobrazení a dále je doporučeno upravit podstruktury skrze formulář editace, pokud nejsou základní informace dostačující. Tyto základní informace jsou: zobrazované jméno, unikátní jméno, typ podstruktury a povinnost. Lze zvolit i neatomické typy, nelze však rovnou editovat jejich podstruktury.



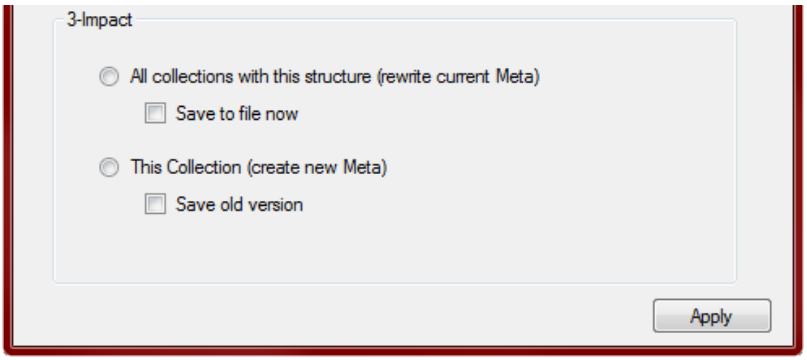
Obrázek 5.7: Výřez z okna pro přídávání struktur: druhý krok

Třetí krok slouží pro zvolení umístění struktury do existujícího stromu. V jediném poli tohoto stromu se vybere struktura jejíž má být vkládaná struktura podstrukturou. V tomto poli jsou uvedeny všechny struktury a podstruktury náležící k načteným datům.



Obrázek 5.8: Výřez z okna pro přídávání struktur: třetí krok

V posledním kroku se zvolí, na která data má mít tato změna metadat vliv. Jsou zde dvě možnosti. Výběr první možnosti způsobí, že se tato změna zapíše do načtených metadat a tím bude mít vliv na všechna data, kterým jsou stávající metadata přiřazena. Druhá možnost způsobí, že se vytvoří nový soubor metadat, který se přiřadí aktuálně otevřené kolekci a všem ostatním datům, které měly doposud stejná metadata jako ta struktura, kterou editujeme zůstanou původní metadata. Dále zde má uživatel možnost uložit ihned metadata do externího souboru.

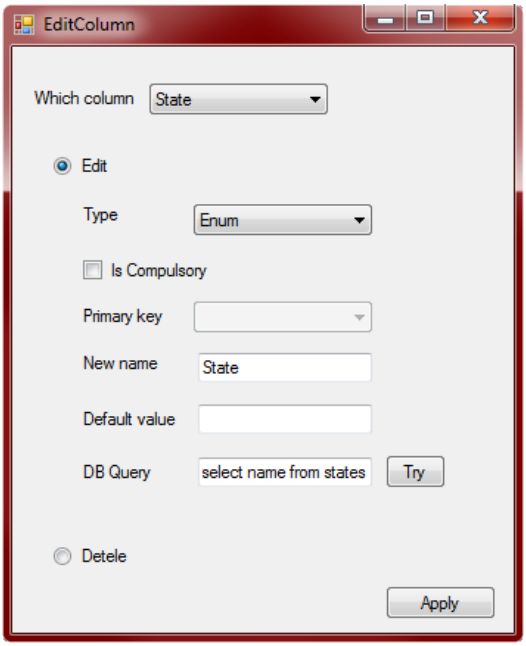


Obrázek 5.9: Výřez z okna pro přídávání struktur: poslední krok

5.2.4 Úprava a mazání metadat

Poté co uživatel vloží do metadat novou strukturu má možnost ji upravit pomocí formuláře vyvolaného po stisknutí tlačítka "Edit Column"vlevo dole v hlavním okně aplikace. Tento formulář umožňuje modifikovat jakoukoli strukturu z těch, které jsou v kolekci metadata. Prvním polem a také prvním krokem uživatele je zvolení struktury, které se budou úpravy týkat. Tuto položku může uživatel buď celou smazat nebo jí upravit, což rozhodne zaškrtnutím jedné komponenty typu RadioButton.

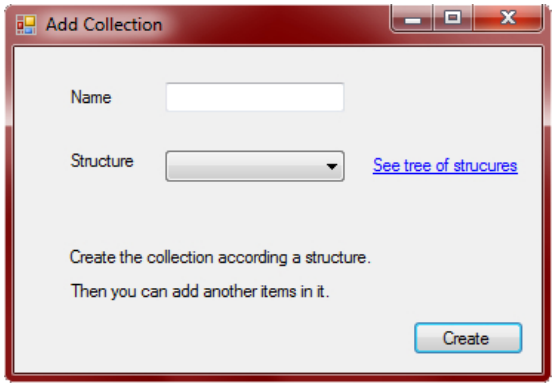
V režimu úpravy může uživatel nastavit všechny atributy struktury. Některá pole budou uživateli znepřístupněna, podle výběru typu struktury, tak aby nedošlo k nekonzistanci dat nebo zmatení. Například kdyby se uživatel snažil přiřadit struktuře, která je listem stromu a nese pouze řetězec, primární klíč. V případě, že uživatel pracuje se strukturou typu "Enum", má umožněno zapisovat do textového pole "DB Query"SQL dotaz a tlačítkem "Try"si může ověřit jeho funkčnost a jím vracené výsledky, které se zobrazí do nového okna jako seznam. Veškeré změny uživatel uloží tlačítkem "Apply"v pravém dolním rohu.



Obrázek 5.10: Okno pro editování stávajících struktur v metadatech

5.2.5 Vytvoření nové kolekce

Vytváření nové kolekce probíhá v samostatném formuláři. Tento formulář po uživateli vyžaduje pouze dvě informace. Zaprvé je to zadání názvu kolekce, který může být naprosto libovolný a nevztahují se na něj ani žádná omezení kvůli unikátnosti neboť otevřená kolekce může být v běhu aplikace pouze jedna. Pod unikátním názvem se tato kolekce uloží až při exportování, kde je její pojmenování plně v režii uživatele. Dále vyžaduje výběr struktury z načtených metadat, podle kterých se budou prvky této kolekce řídit. Pro lepší orientaci si může uživatel zobrazit hierarchii struktur kliknutím na odkaz pro zobrazování dostupných metadat přímo v tomto formuláři umístěný vedle komponenty, kterou bude provádět výběr.



Obrázek 5.11: Okno apliakce pro vytváření nové kolekce

Kapitola 6

Implementace

Implementovaná aplikace je zpracována objektově orientovaným přístupem a velmi důležitým momentem implementace je využití rekurzivního volání metod. Využití rekurze je velmi výhodné pro přístup k hierarchicky zanořeným datům a v apliakci je využíván jak pro zobrazování veškerých formulářů, tak pro čtení dat z uživatelských vstupů.

Nejdůležitější body implementace a podorobnosti o zvolených technologiích jsou popsány v následujícícih kapitolách.

6.1 Použité technologie

Pro vývoj implementovaného generátoru byly použity následující technologie a nástroje: platforma .NET verze 4.5, jazyk C# Microsoft SQL Server 2012 a vývojové prostředí Visual Studio 2012. Program byl vyvíjen pod operačním systémem Microsoft Windows 7 Professional. Pro grafické uživatelské rozhraní byly zvoleno API Windows Forms.

Tato varianta byla vybrána také z důvodu možnosti využití výsledného produktu pro možnost prototypování workflow systému v komerčním prostředí, kde jsou technologie Microsoft Windows využívány.

6.1.1 .NET

.NET je označení pro vývojovou a exekuční platformu vyvinutou pro prostředí MS Windows. Možnost pracovat nad platformou .NET existuje i pro uživatele jiných operačních systémů. Kvůli této možnosti vzniklo množství projektů, které se soustředí na přenositelnost či vývoj nástrojů pro "unix like"operační systémy pro práci nad platformou .NET, které jsou svobodné. Mezi než patří například nezávislé projekty Mono či dotGNU.

Součástí systému MS Windows 7 je pouze verze .NET 3.5. Použitou verzi 4.5 je potřeba doinstalovat a vyžaduje na operačním systému aplikovaný service pack 1.

Významným faktem o platformě .NET je unifikování některých vlastností více programovacích jazyků tak, aby několika z těchto jazyků mohla být bez překážek psána jedna a tatáž aplikace. Pro tuto unifikaci se využívá Společné běhové prostředí (Common Language Runtime či CLR) s Společného systému typů (Common Type System či CTS) dále Společné specifikace jazyků (Common Language Specification či CLS) a Knihovny základních tříd (Base Class Library či BCL). Mezi jazyky platformy .NET patří například: C#, C++, Visual Basic nebo F#.

CLR je virtuální stroj čili prostředí ve kterém běží již přeložený program, sama také vykonává kód tak, že překládá předzpracovaný kód do podoby metadat ve specifickém

jazyce MSIL do strojového kódu. Tento překlad se děje až za chodu programu a proto se tyto kompilátory nazývají JIT (Just-in-time). Dále se stará o záležitosti jako je správa paměti (ochrana neoprávněných přístupů, garbage collector…) a řešení výjimek.

CTS udává základní typy (třída, struktura, rozhraní, výčet, delegát), které mohou používat všechny jazyky na platformě .NET. Jeden konkrétní jazyk však nemusí využít všechny možnosti.

CLS je množina syntaktických a sémantických programových rysů, které zajistí přenositelnost objektů z jednoho jazyka platformy .NET do druhého. Společně s CTS už dokáží zajistit možnost vývoje jedné aplikace ve více jazycích.

BCL je podmnožinou FCL (Framework Class Library).

Možnost vypracovat jednu aplikaci ve více programovacích jazycích byla dalším z důvodů (a to velmi důležitým při práci s tak komplexním systémem jako je workflow systém) výběru těchto technologií. Dalším důvodem je snadná úprava desktopové aplikace na aplikaci s webovým rozhraním, kde při využití technologie ASP .NET může zůstat jádro programu stejné, pouze se modifikuje uživatelské rozhraní [11].

6.1.2 Implementační jazyk C#

Jazyk C# byl vyvinut speciálně pro platformu .NET společností Microsoft. Současný standart pro tento jazyk byl formulován v roce 2006 [5], jedná se již však o čtvrtou verzi. Tento jazyk vychází z jazyka C++ (tedy syntakticky z jazyka C) a velmi připomíná jazyk Java. Jedná se o vysokoúrovňový objektově orientovaný jazyk. Přesněji jazyk staticky typovaný, poskytující pouze jednoduchou dědičnost a hybridně objektový: obsahuje primitivní datové typy, které lze skládat do homogenních nebo heterogenních struktur. Takovouto strukturu pak můžeme nazvat třídou. Obsahuje jak primitivní datové typy (atributy), tak i metody. Všechny třídy jsou pak v případě jazyka C# odvozeny od jedné kořenové třídy, která je jejich předkem.[13]

Jazyk C# byl vybrán především pro svou univerzálnost a poskytovanou velkou programátorskou efektivnost ve spojení s vývojovým a exekučním prostředním Visual Studio 2012. Také pro široké možnosti práce s grafickým uživatelským rozhraním, ať už pro desktopové aplikace nebo pro vývoj webového grafického rozhraní s nástrojem ASP .NET.

6.1.3 Framework pro grafické uživatelské rozhraní

Při výběru frameworku pro tvorbu grafického uživatelského rozhraní bylo rozhodováno mezi dvěma variantami dostupnými pro jazyk C#: Window Forms a Windows Presentation Foundation.

Windows Forms (dále jen WF) používá prezentaci grafických komponent v kódu jazyce C# jako instance jednotlivých tříd pro každý druh grafické komponenty. Takovéto třídy mají své atributy a metody, pomocí kterých se komponentám nastavují vlastnosti, akce nad nimi prováděné a události na ně navazující. Na platformě .NET je dostupné od verze 1.0.

Windows Presentation Foundation (dále jen WPF) využívá zápisu uživatelského rozhraní ve značkovacím jazyce XAML upraveném pro potřeby frameworku. Všechny grafické komponenty tvoří strom, který má jako kořen zapouzdřující prvek, celého formuláře. Tento strom zahrnuje také vlastnosti jednotlivých prvků, které jsou reprezentovány jako další jeho úrovně. WPF můžeme využít na platformě .NET od verze 3.0 a poskytuje rozšířené grafické možnosti pro uživatelské formuláře, především estetického rázu, jako jsou: efekty jako

průhlednost či záření, práci s multimédii nebo animace. Pro zvládnutí těchto možností běží WPF nad vrstvou DirectX a zobrazení se zpracovává přímo na grafické kartě.

Pro implementovanou aplikaci byla zvolena varianta Windows Forms neboť jí poskytované vlastnosti jsou dostačující. Speciální grafické efekty, které poskytují WPF, nebudou pro tuto aplikaci zapotřebí, neboť jejím účelem není po estetické stránce nadchnout uživatele, ale prakticky a rychle zobrazit úpravy, provedené kvalifikovaným uživatelem nebo programátorem, při prototypování.

6.1.4 Microsoft SQL Server

MS SQL server byl vybrán jako technologie umožňující práci s relační databází, kompatibilní s ostatními technologiemi MS. Byla použita jeho poslední dostupná verze, která vyžaduje na operačním systému Windows 7 nainstalovaný Service Pack 1.

6.1.5 XML

Jazyk XML byl zvolen pro ukládání hierarchických dat do souborů. Jedná se o rozšiřitelný značkovací jazyk pomocí něhož můžou být zaznamenávána libovolně větvená data, která nemají kruhovou závislost. Musí se tedy jednat o stromy. V Implementované aplikaci jsou do formátu XML serializovány celé třídy reprezentující data i metadata programu. [7]

Dalším z důvodů zvolení jazyka XML byl fakt, že je v oblasti workflow systémů velmi významnou technologií. Především pro přenos a ukládání dat. Mimo jiné je základem pro několik jazyků, které navrhla instituce Workflow Management Coalition. Jedná se například o jazyky XPDL (XML Process Definition Language) BPML (Business Process Modeling Language) a jeho následovník BPEL4WS (Business Process Execution Language for Web Services) pro definici procesu ve workflow systému. Jelikož workflow systémy nemusejí běžet nutně na jednom stroji, tedy existuje zde síťová komunikace je XML také formátem pro výměnu dat určeným protokolem SOAP (Simple Object Access Protocol), protokolem pro výměnu zpráv po počítačové síti.

6.2 Validace vstupů

Veškeré uživatelské vstupy do formulářů, se kterými se pracuje musí být validovány, aby přijatá data mohla být dále automaticky zpracovávána. Validace dat je závislá na jejich typu, který je uložen v atributu struktury "Type". Tento atribut je typi DataType, což je výčtový typ, díky němuž je možný výběr vypů omezen pouze na: textové řetězce, čísla, výčty, časové údaje a struktury - označení pro strukturu, která se dále větví. Podle tohoto atributu se při tištění formuláře na hlavní okno aplikace vybírají grafiké komponenty, které budou použity pro uživatelský vstup dané položky. Způsob validace jak pak zvislý na zvolené grafiké komponentě.

6.2.1 Textové řetězce

V případě textových řetězců, u kterých není potřeba vstupy nijak omezovat je použita komponenta TextBox, ze které se při uložení čte hodnota jejího atributu "Text", který je typu řetězec. Do tohoto komponentu je možno zadávat všechny alfanumerické znaky klávesnice.

6.2.2 Čísla

Pro čísla byla navržena využita vlastní třída "NumberBox", který dědí za třídy "TextBox"všechny vlastnosti kromě metody "OnKeyPress", kterou přetěžuje. Tělo této metody pak při každém stisknutí klávesy (pokusu o zápis do vstupu, který je instací třídy NumberBox) zkontroluje jestli je zasílaný znak číslicí či desetinnou tečkou nebo čárkou. Pokud tomu tak není přiřadí do atributu "Handled"logickou hodnotu true a po ukončení této funkce již nedojde ke zpracování tohoto zaslaného znaku a čeká se na signál další uživatelem stištěné klávesy. Pokud příchozí znak je číslicí tečkou nebo čárkou, dojde k jeho dalšímu zpracování. V případě číslice dojde přímo k jeho zobrazení v poli uživatelského vstupu. V případě desetinné tečky a čárky dojde ke kontrole, zda vstupní pole již neobsahuje některý znak oddělující celou část čísla od desetinné. Pokud obsahuje, opět se nastaví atribut "Handled"na hodnotu true a znak se dále nezpracovává. V opačném případě se zobrazí do pole vstupu.

Načítat čísla jde samozřejmě i ze vstupu třídy TextBox pomocí metod pro čtení čísel z textových řetězců v kombinaci s přetypováním. Pro tento případ můžou být použity například metody ToInt32 pro přetypování a metoda Parse pro čtení dat.

6.2.3 Výčty

Výčtové vstupy jsou relalizovány pomocí komponent podle třídy ComboBox. Pokud je uživateli povoleno vkládat vlastní varianty nad rámec těch, které jsou nabídnuty v rozbalovacím seznamu, řídí se tyto vstupy stejnými pravidly jako vstupy pro textové řetězce. V případě, že uživatel nemá mít možnost vložit vlastní vstup do pole ComboBox, je jeho atribut DropDownStyle nastaven na hodnotu DropDownList. Takový to ComboBox se pak jeví jako pevná šedá komponenta, která může zobrazit možnosti, ale nejde do ní nic vepisovat.

6.2.4 Časové údaje

Uživatelský vstup časových údajů je jedním z případů vstupů, který je velmi nutné validovat, pokud má být dále automaticky zpracován, neboť pro zápis času a dat existuje příliš mnoho variant a formátů.Vstupní pole, do kterých má uživatel zadávat časové údaje jsou realizovány komponentami třídy DateTimePicker, která kromě zajištění, že do nich nepůjde zapsat nic, co neodpovídá šabloně pro časový údaj, poskytuje uživateli také možnost zobrazení kalendáře, ze kterého vybere datum a tak se nemusí zabývat formátem, ve kterém časový údaj zadá. Kalendář napomůže také například v situaci, kdy si uživatel není jist v jakém pořadí jsou zadávány položky měsíc a den v případě, kdy měsíc je označen pouze číselně a ne svým názvem.

6.3 Serializace

Pro serializaci dat i metadat byla zvolena třída DataContractSerializer, která je na platformě .NET dostupná od verze 3.0. Tato možnost byla po zvážení upřednostněna před starším způsobem řešení serializace pomocí XmlSerializer. V zpracovávaném případě, kdy vždy z jedné třídy serializujeme všechny atributy, by se mohl jevit jako lepší přístup XmlSerializer. XmlSerialier vždy serializuje z dané třídy vše, co není označeno atributem určujícím, že tento atribut má být ignorován. Naopak u DataContractSeliarizer musí být u každého atributu, který má být serializován, uvedena značka (atribut) "DataMember". Rozhodujícím rozdílem však byl fakt, že DataContractSerializer je schopen bez jakýchkoli úprav

serializovat i Dynamické slovníky (DynamicDictionary), s jejichž hash kódy má XmlSerialize problém. Pro XmlSerializer se tento problém řeší například pomocnou třídou, která nahrazuje dvojice ¡klíč, honota¿ ve slovníku a použitím obyčejného seznamu instancí těchto hodnot [6].

Serializace je v implementované aplikaci využita při exportování kolekcí dat a metadat.Výše zmíněným způsobem jsou serializovány třídy Structure a Metadata, které obě obsahují atribut typu dynamický slovník. Serializace třídy Structure dostačuje i pro serializaci celé kolekce zpracovávaných dat, neboť v jejím případě se serializuje a exportuje pouze její atribut data, který je kolekcí tedy kořenem stromu dat. Reprezentace pomocí XML se výborně hodí pro ukládání hierarchických dat, v případě exportování dynamických slovníků si však Serializer vypomáhá pomocí XML Schema, což je vidět v následujícím kódu, ukázce exportovaných dat, úryvku z jedné kolekce.

V začátku dokumentu jde vidět, že exportovaná třída obsahuje jeden atribut typy Structure. Jedná se o kolekci jménem zaměstnanci, jejíž záznamy se řídí podle metadat pojmenovaných Person.

```
<?xml version="1.0"?>
-<Structure xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.datacontract.org/2004/07/BP">
```

Protože se v serializované třídě vyskytují atributy typu DynamicDictionary a Object je potřeba využít kromě čistého XML také jeho propojení s XML Schema. Dále následují atributy kolekce. Jedním z nich je atribut Items, který nese údaje o jednom zaměstnanci v tomto úryvku tyto údaje představují jednu hodnotu a to jméno.

```
<DBQuery i:nil="true"/>
-<Items xmlns:a="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
-<a:KeyValueOfstringanyType>
<a:Key>8808253695</a:Key>
-<a:Value i:type="Structure">
<DBQuery i:nil="true"/>
```

Následuje pole informací o jednotlivém zaměstnanci s jedním prvkem - jménem. Vidíme začátek pole Items a XSD notaci pro klíče a prvky tohoto slovníku. Prvek defValue je uložen s výpomocí XML Schema z důvodu, že je definován jako typ Object, ale nese hodnotu typu String.

```
-<Items>
-<a:KeyValueOfstringanyType>
<a:Key>1</a:Key>
-<a:Value i:type="Structure">
<DBQuery i:nil="true"/>
<Items/>
<PK/>
<cmopulsory>false</cmopulsory>
<defValue i:type="b:string" xmlns:b="http://www.w3.org/2001/XMLSchema"/>
<metaType>Leaf</metaType>
<name>Given name</name>
<structDefName>1</structDefName>
<type>String</type>
<value i:type="b:string" xmlns:b="http://www.w3.org/2001/XMLSchema">
```

```
František</value>
</a:Value>
</a:KeyValueOfstringanyType>
</Items>
```

Zde končí údaje o jednom zaměstnanci. Na následujícím řádku je ještě uloženo pod jakým primárním klíčem vystupuje v kolekci Zaměstnanci. A následují atributy struktury, která reprezentuje jeden záznam v kolekci Zaměstnanci.

```
<PK>8808253695</PK>
<cmopulsory>false</cmopulsory>
<defValue i:type="b:string" xmlns:b="http://www.w3.org/2001/XMLSchema"/>
<metaType>Str</metaType><name>Person</name>
<structDefName>Person</structDefName>
<type>String</type>
<value i:type="b:string" xmlns:b="http://www.w3.org/2001/XMLSchema"/>
</a:Value>
</a:KeyValueOfstringanyType>
</Items>
```

Konec informací o jednom konkrétním zaměstnanci. Začínají atributy samotné kolekce Zaměstnanci, kde je v položce PK uvedeno která informace o jednotlivých zaměstnancích bude klíčem pro kolekci (slovník) zaměstnanců.

```
<PK>Personal number</PK>
<cmopulsory>false</cmopulsory>
<defValue xmlns:a="http://www.w3.org/2001/XMLSchema" i:type="a:string"/>
<metaType i:nil="true"/>
<name>Zaměstnanci</name>
<structDefName>Person</structDefName>
<type>String</type>
<value xmlns:a="http://www.w3.org/2001/XMLSchema" i:type="a:string"/>
</Structure>
```

Kapitola 7

Testování

Pro účely testování byla navržena sada testů, které mají ověřit, zda aplikace korektně generuje formuláře, ukládá data z jejich vstupů, poskytuje znovupoužitelné exportované soubory a zda jsou funkční její části poskytující možnosti k úpravě metadat. Dále byla aplikace předložena uživatelům, aby zhodnotili její uživatelské rozhraní a míru složitosti a účelnosti jejího použití.

7.1 Testy

Aplikace byla několikrát testována sadou testů uvedených v příloze C. Vývoj a testování probíhali iterativně.

Všechny testy (kromě exportu) testují možnost provést tento úkol a vizualizaci jeho výsledku do hlavního okna aplikace.

7.2 Uživatelský průzkum

Aplikace a návod k jejímu použití byl předložen dvaceti uživatelů, kteří splňovali požadavek na minimálně středoškolské vzdělání technického typu. V této množině byli 2 lidé do věku 18 let, 8 Lldí ve věku 18 až 24 let, 5 lidí mezi 25 a 30 lety, 3 ve věku od 31 do 40 let a 2 lidé starší 40 let.

Uživatelům byly předloženy následující tvrzení, které měli ohodnotit podle toho jak s nimi souhlasí, výběrem čísla ze stupnice 1 až 5. 1 - naprosto souhlasím, 5 - neprosto nesouhlasím. Za každým z předložených tvrzení je v závorce uvedeno průměrné uživatelské hodnocení. Každý uživatel měl možnost doplnit k hodnocení libovolný komentář.

- Ovládání aplikace je jednoduché. (1,8)
- Generované formuláře jsou přehledné. (1,2)
- Grafické uživatelské rozhraní je příjemné. (2.6)
- Upravování metadat je jednoduché. (3.6)
- Upravování dat je jednoduché. (1,9)

Poznámky uživatelů, které se objevily opakovaně:

- Nepohodlné vybírání toho, kterou strukturu upravuji. (V okně Edit Column a Add Column)
- Neintuitivní upravování strutkru - nepřístupnost k úplravě podrstruktur dané struktury v daném formuláři a nutnost uložit a otevřít druhý formulář.
- Nevratnost změn, pokud nebyla ručně exportována záloha do souboru.

Kapitola 8

Možná rozšíření

Jak již bylo naznačeno v teoretických kapitolách, kde je zmiňována možnosto jazyka C# pracovat s více grafickými frameworky, je prvním navrhovaným rozšířením vznik webové verze aplikace, tedy přepracování jajího uživatelského rozhraní v technologii ASP. NET a umožnění práce se soubory v počítačové síti, například s využitíím formýtu JSON.

Aby aplikace mohla uživateli, u kterého je prováděno prototypování více způsoby, znázornit možnost práce s daty workflow je navrženo rozšíření o více způsobů zobrazení formulářů. Prvním krokem tohoto rozšíření je dopracování možnosti zobrazování nazvané New Window View tak, aby poskytovala všechny možnosti úpravy dat i metadat. Dále jsou navrženy možnosti zobrazování, pracovně pojmenované +view a nonEditView. Metoda +view by zobrazovala v hlavním okně aplikace pouze jednu úroveň zanoření a položky, které by měly podstruktury, by byly pomocí symbolu + rozrolovatelné a zobrazitelné. Metoda nonEditView by zakázala editaci záznamu, pole uživatelských vstupů a jejich hodnoty by zobrazila pouze jako text, aby do nich nemohl uživatel zasahovat.

Dalším rozšířením aplikace by mohla být možnost porovnávání jednotlivých záznamů v kolekci nebo prohlížení jiných dat a metadat než těch, se kterými se aktuálně pracuje. Šlo by o možnost pracovat v aplikaci zároveň s více kolekcemi dat a metadat. Takováto změna by nebyla náročná, pouze by do singletonů pro data a metadata jednoduché položky nahradila poli těchto položek a dále by se muselo upravit pouze chování GUI.

Při úpravách metadat, hodnotili uživatelé jako nepohodlný způsob vybírání struktury, které se budou změny týkat. Návrhem je vybírat strukturu k úpravě ještě před stisknutím tlačítka, které otevře dialog Edit Column a to například takovým způsobem, že jí uživatel označí v grafickém rozhraní kliknutím na ni, ta se pak graficky zvýrazní (například rámečkem), aby byla odlišitelná a jasně rozpoznatelná, že je vybrána.

Další připomínkou k uživatelské přívětivosti úpravy struktur byla neintuitivnost nutnosti vrátit se do hlavního okna a otevřít jiný dialog pro přidání položek do struktury, která byla právě upravována v dialogu Edit Column. Řešením, které by mělo zjednodušit práci by bylo, přidat do dialogu Edit Column tlačítko, které by otevřelo dialog pro přidávání položek. Toto tlačítko by bylo přístupné pouze u typu struktury struktura nebo kolekce.

Dalším návrhem pro usnadnění demonstrování změn na datech a metadatech je možnost vrácení posledních několika kroků a jejich obnovení. Navrhované řešení je přidat do horní lišty aplikace menu Edit, které bude obsahovat položky Undo a Redo, tak jak je uživatelé znají z aplikací typu textových editorů a podobně.

Kapitola 9

Závěr

Vrámci této práce byl navržen a impelementován a testován program, který na základě metadat uložených v souboru dokáže zobrazovat uživatelské formuláře. Tyto formuláře dokáže upravovat na úrovni metadat i dat. Aplikace validuje uživatelské vstupy podle nastavení v metadatech a včechny zpracovávané informace (jak data tak metadata) dokáže exportovat do XML souboru.

Na základě předložení aplikace uživatelům a jejich připomínek a podle zjištěných souvislostí při návrhu, implementaci a testování aplikace bylo navrženo několik možných rozšíření do budoucna.

Aplikace může být využita jako modul pro připravovaný workflow systém, avšak před jejím itegrací je potřeba opravit dva zjištěné nedostatky při testování a zvážit možnosti propojení s konkrétním systémem a doimplementování automatického načítání příchozích souborů s daty a metadaty a upravit exportování tak, aby dobře navazovalo na konkrétní workflow systém.

Literatura

[1] Antonín Carda, Renáta Kunstová: *Workflow, Řízení firemních procesů*. Grada Publishing, 2001, iSBN 80-247-0200-2.

[2] Charles Petzold: *Programování Microsoft Windows Forms v jazyce C#*. Computer Press, 2006, iSBN 80-251-1058-3.

[3] Coalition, W. M.: Workflow Management Coalition Terminology & Glosary [online]. `www.wfmc.org/Download-document/WFMC-TC-1011-Ver-3-Terminology-and-Glossary-English.` 1999-02 [cit. 2013-04-26].

[4] D. Crockford: RFC 4627 [online]. `http://tools.ietf.org/html/rfc4627`, 2006 [cit. 2013-26-04].

[5] Ecma International: Standard ecma-334 [online]. `www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf`, 2006-06 [cit. 2013-04-28].

[6] Microsoft: MSDN Library. `msdn.microsoft.com/library/default.aspx`, 2013 [cit. 2013-09-05].

[7] Neil Bradley: *XML - kompletní průvodce*. Grada Publishing, 2000, iSBN 80-716-9949-7.

[8] Nick Russell, D. E. W. M. v. d. A., Arthur H.M. ter Hofstede: WORKFLOW DATA PATTERNS [online]. `www.workflowpatterns.com/documentation/documents/data_patterns%20BETA%20TR.pdf`, 2004 [cit. 2013-04-27].

[9] Prof. Ing. Tomáš Hruška, CSc. Ing. Vojtěch Mates: *Pokročilé Informační systémy (PIS), Workflow, Studijní opora*. FIT VUT Brno, 2012-02.

[10] Prof. Ing. Tomáš Hruška, CSc. Ing. Zbyněk Křivka: Informační systémy (IIS,PIS), Pojem informačního systém, Data, Procesy, Transakce, Studijní opora. 2012-02.

[11] Slavoj Písek: *ASP.NET (začínáme programova)*. Grada Publishing, 2003, iSBN 80-247-0526-5.

[12] Sotiris Zigiaris, M.: BUSINESS PROCESS RE–ENGINEERING BPR [online]. `www.adi.pt/docs/innoregio_bpr-en.pdf`, 2000-01 [cit. 2013-04-26].

[13] Zbyněk Křivka, Dušan Kolář: *Principy programovacích jazyků a objektově orientovaného programování, IPP II, Studijní opora*. FIT VUT Brno, 2008-02.

Příloha A

Obsah CD

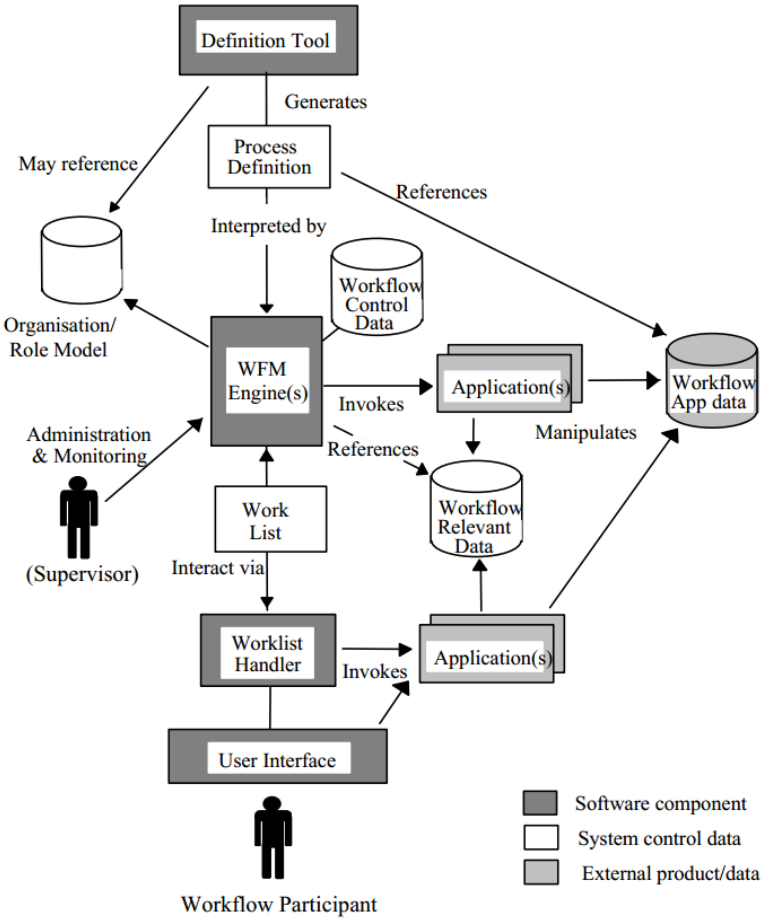
`bin`	skompilavaný program
`export`	soubory exportované testy
`src`	zdrojové kódy
`src_zprava`	zdrojový kód technické zprávy
`bakalarska_prace.pdf`	elektronická verze technické zprávy
`doc.txt`	programová dokumentace

Příloha B

Obrázky

Construct	Staffware	MQSeries	FLOWer	COSA	XPDL	BPEL4WS
string	•	•	•	•	•	•
integer	•	•	•	•	•	•
float	•	•	•	•	•	•
boolean			•	•	•	•
date	•		•	•	•	•
time	•		•	•		•
document/reference	•		•	•	•	•
enumeration					•	•
composition	•	•	•		•	•
array		•	•		•	
set						•

Obrázek B.1: Datové typy používané v jednotlivých workflow jádrech [8, str. 4]



Obrázek B.2: Struktura workflow systému[8, str. 4]

Příloha C

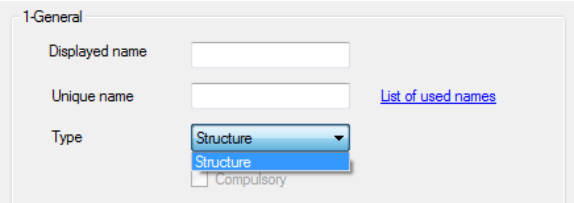
Navržené testy a jejich výsledky

Test 1: Vytvoření nových metadat bez vícenásobně zanořených struktur.

Postup:
V menu File položka New a její volba Metadata otevřít nové okno. V otevřeném formuláři "AddItem"vyplnit zpřístupněná pole prvního boxu. V boxu 1a přidat několik podstruktur typu String, Number a Date. Ve třetím boxu vybrat první volbu a aplikovat.

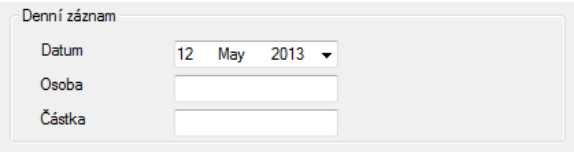
Očekávaný výsledek:
Aplikace by měla zamezit vytvořit do prázdných metadat do první úrovně jiný prvek, než prvek typu structure a vykreslit do hlavního okna Groupbox s několika položkami. Po otevření okna "AddItem"vymazání aktuálních metadat. Po vyplnění a odeslání formuláře okna "AddItem"zobrazení prázdného formuláře obsahující Groupbox s několika položkám typu String, Number nebo Date.

Výsledek:
Aplikace zamezila výběru jiného typu než Structure odebráním ostatních možností z prvku ComboBox pro výběr typu.



Obrázek C.1: Omezený výběr v poli Type

Grafický výstup v hlavním okně byl správný. Zobrazily se pole typu DateTimePicker, TextBox a NumberBox a jejich popisy.



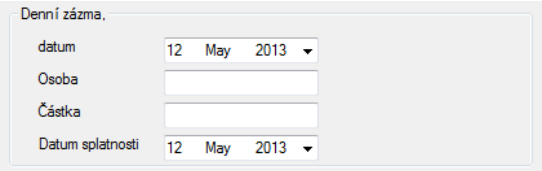
Obrázek C.2: GUI Test 1

Test 2: Přidání struktury do metadat.

Postup:
Pomocí tlačítka Add Column přidat novou strukturu, jako podstrukturu stávající struktury první úrovně do metadat. Vyplnit pole prvního boxu, jako typ přidávané struktury zvolit postupně Date, String a Number. Ve třetím boxu zvolit 1. možnost. Aplikovat.

Očekávaný výsledek:
Znovuzobrazení prázdného formuláře podle stávajících metadat s doplněním o nově přidanou strukturu. Metadata stále nejsou vázána k žádným konkrétním kolekcím.

Výsledek:
Korektně se zobrazily přidané položky ve všech třech variantách testu. Na následujícím obrázku je znázorněno úspěšné zobrazení první varianty - s přidaným datem.



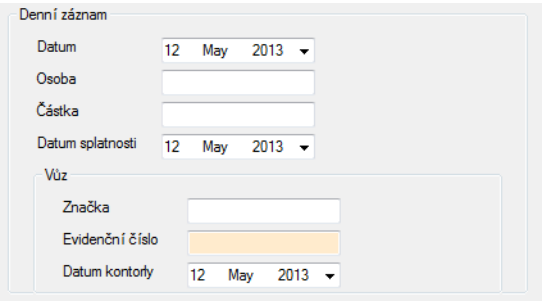
Obrázek C.3: GUI Test 2

Test 3: Přidání zanořené struktury do metadat a vyznačení povinného pole

Postup:
Postup tohoto testu je obdobný jako Test 2. Rozdíl bude učiněn ve výběru typu přidávané struktury. Tato struktura bude typu "Structure". V zpřístupněném boxu 2 pak přidáme 3 hodnoty každou jiného typu (String, Number, Date). Položku typu Number označit jako povinnou označením volby "Comp".

Očekávaný výsledek:
Znovuzobrazení prázdného formuláře podle stávajících metadat s doplněním o nově přidanou strukturu, která bude zobrazena jako GroupBox s třemi položkami typu String, Number a Date. Položka Typu Number, která je povinná bude zvýrazněna žlutým podbarvením. Metadata stále nejsou vázána k žádným konkrétním kolekcím.

Výsledek:
Korektně se zobrazily přidané položky včetně vyznačení povinného pole. Na následujícím obrázku je znázorněno úspěšné zobrazení celé přidané struktury.



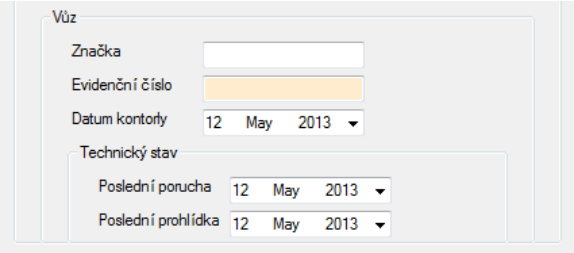
Obrázek C.4: GUI Test 3

Test 4: Více úrovní zanoření

Postup:
Aplikovat postup z testu 3 na zanořenou strukturu (vybrat ji v poli "In Structure").

Očekávaný výsledek:
Zobrazení vnořeného GrouBoxu v Groupboxu vloženém v testu 3.

Výsledek:
Korektně se zobrazily přidané položky včetně vyznačení povinného pole. Na následujícím obrázku je znázorněno úspěšné zobrazení celé přidané struktury.



Obrázek C.5: GUI Test 4

Test 5: Vícenásobné použití jedné struktury

Postup:
Přidat pomocí okna Add Item strukturu, která je již jednou v metadatech použita.

Očekávaný výsledek:
Zobrazení dvou struktur se stejnými prvky v jedné nadřazené struktuře. Například adresa trvalého pobytu a adresa přechodná.

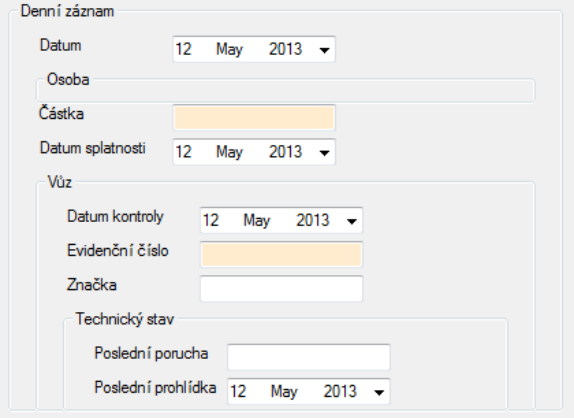
Výsledek:
Při využití struktury, která není a atomická a je využita výše v hierarchii struktur dochází k cyklení. Tato funkčnost byla z aplikace po testování odebrána a takovouto duplikovanou strukturu musí uživatel vytvořit ručně s jiným unikátním jménem. Pro zrychlení ji může nakopírovat v exportovaném XML a změnit unikátní jméno a doplnit odkaz na něj.

Test 6: Upravení stávající struktury v medatatech

Postup:
Upravit pomocí okna Edit Column jakoukoli strukturu v hierarchii, která je již jednou v metadatech použita. Vyzkoušet na všech úrovních zanoření změnu všech zpřístupněných atributu struktury. Zpřístupnění je vázáno na typ struktury.

Očekávaný výsledek:
Zobrazení pozměněných metadat bez narušení jejich hierarchické struktury.

Výsledek:
Aplikac nepovolila všechny kombinace změn nezávisle na úrovni zanoření. Na následujícím obrázku je snímek varianty, kde byla nahrazena položka typu String strukturou, do které se můžou později přidat položky. Byla změněna povinnost jedné položky. Dvě položky se vyměnily navzájem a z položky typu Date byla udělána položka typu string.



Obrázek C.6: GUI Test 6

Test 7: Smazání struktury v metadatech

Postup:
Pomocí okna Edit Column a výběru varianty Delete. Smazat struktruru.

Očekávaný výsledek:
Ostranění struktury z kolekce metadata odstranění všech odkazů na tuto strukturu.

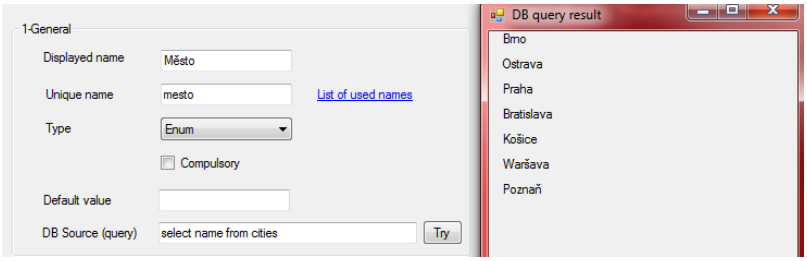
Výsledek:
Struktura byla úspěšne smazána včetně všech odkazů nani. Při mazání struktury z metadat, které jsou již aplikována na konkrétní kolekci může dojít k nesprávnému zobrazení dat kolekce.

Test 8: Položka typu Enum v metadatech - provázání s relační databází

Postup:
Pomocí okna Add Item přidat novou strukturu typu Enum. U této struktury vyplnit dotaz do relační databáze a ten otestovat pomocí tlačítka "Try".

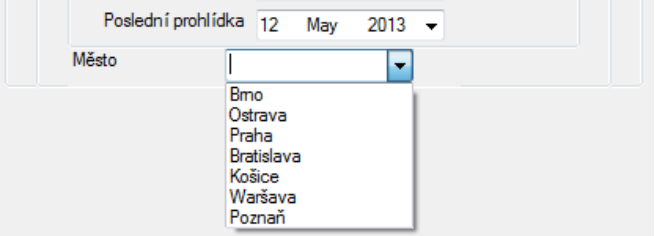
Očekávaný výsledek:
Po vybrání typu Enum se zpřístupní možnost vyplnit SQL dotaz. Po jeho otestování tlačítkem "Try"se zobrazí nové okno se seznamem zobrazující výsledky tohoto dotazu. V hlavním okně aplikace se znovu zobrazí prázdný formulář s přidaným prvkem typu ComboBox, jehož položky budou naplněny výsledkem SQL dotazu.

Výsledek:
Zpřístupnění pole pro SQL dotaz proběhlo úspěšně. V případě správně zadaného dotazu se zobrazil seznam hodnot v novém okně, v opačném případě informace o nevaliditě dotazu.



Obrázek C.7: GUI Test 8 - přidávání

V hlavním okně se prvek typu ComboBox zobrazil korektně.



Obrázek C.8: GUI Test 8 - výsledek

Test 9: Vytvoření nové kolekce

Postup:
Pomocí nabídky File její položky New a volby Collection se otevře nové okno pro vytváření kolekce, kde se napíše její název a vybere, podle které části metadat se bude řídit.

Očekávaný výsledek:
Zobrazení okna pro vytváření kolekcí. Znovunačtení formuláře v hlavním okně pole, kde se zobrazí už jen ty položky, které odpovídají zvolené části metadat.

Výsledek:
Otevření nového oka proběhlo úspěšně. Obsah hlavního okna se správně překreslil pro všechny varianty výběru částí metadat, podle kterých se má řídit kolekce.

Test 10: Uložení dat do kolekce

Postup:
Do připravené vytvořené kolekce vepsat údaje a potvrdit je tlačítkem "Save entry to collection".

Očekávaný výsledek:
Objeví se hláška "Saved"a v poli PK (nahoře ve formuláři) bude dostupný jako jedna z možností rozbalovací nabídky primární klíč záznamu.

Výsledek:
Funguje korektně pouze pro kolekce, které se řídí podle celých metadat tedy včetně jejich kořene. U kolekcí, které se řídí pouze podle části metadat (některé zanořené struktury), uložení neproběhne z důvodu nenalezení primárního klíče záznamu. Chyba vznikla při při-

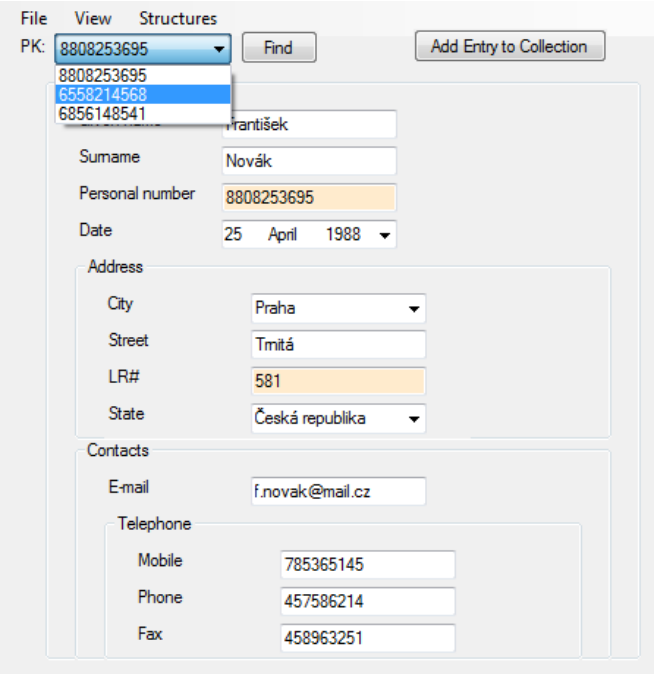
dávání podstruktur, kde se nesprávně přiřazuje název primárního klíče. Tento stav je ošetřen chybovou hláškou "PK not found".

Test 11: Čtení dat z kolekce podle primárního klíče

Postup:
Výběr primárního klíče v poli "PK"a stisknutí tlačítka "Find";

Očekávaný výsledek:
V hlavním okně zůstane načtený formulář, ale obsah jeho vstupních polí se přepíše hodnotami ze zvoleného záznamu.

Výsledek:
Pro struktury, které se řídí celými metadaty je funkční. Pro struktury řídící se podmnožinami metadat nemohlo být testováno z důvodu neúspěšnosti testu 10. Při prvním čtení dat neproběhne čtení korektně - nezobrazí se hodnoty. Je potřeba na tlačítko "Find"kliknout ještě jednou.



Obrázek C.9: GUI Test 10, 11 - ukládání a čtení podle PK

Test 12: Úprava metadat kolekce, která již obsahuje prvky

Vynechán na základě testu 7.
Úpravy metadat, které jsou již propojeny s nějakou kolekcí způsobuje nesprávne zobrazování záznamů kolekce z důvodu španě uložených klíčů. **Test 13: Exportování metadat**

Postup:
Pomocí menu File a jeho položky Export all meta uložit metadata do souboru.

Očekávaný výsledek:

XML soubor s kolekcí všech dostupných metadat.

Výsledek:
Export proběhl úspěšně. Příklad vyexportovaných metadat s názvem Person, je v příloze A jako soubor Person.xml.

Test 14: Exportování kolekce

Postup:
Pomocí menu File a jeho položky Export collection zpracovávanou kolekci do souboru.

Očekávaný výsledek:
XML soubor s kolekcí a všemi jejími záznamy.

Výsledek:
Export proběhl úspěšně. Příklad vyexportovaných dat kolekce zaměstnanců podle metadat s názvem Person je v příloze A jako soubor DvaZaznamyPerson.xml.

Test 15: Importování metadat

Postup:
Pomocí menu File a jeho položky Import all meta načíst do aplikace metadata ze souboru.

Očekávaný výsledek:
Vymazání aktuálních metadat. Přidání nových struktur do silgletonu Metadata a přiřazení jména kořene do atributu root.

Výsledek: Proběhlo v pořádku. Nová metadata se vizualizovala, jdou upravovat a jde podle nich vytvořit kolekce, se kterou se dá dále pracovat.

Test 16: Otevření uložené kolekce

Postup:
Pomocí menu File a jeho položky Open collection načíst ze souboru uloženou kolekci záznamů.

Očekávaný výsledek:
Výzva k uložení stávající kolekce. Přiřazení nové kolekce do singletonu CurrentData a zobrazení k ní příslušejícího formuláře, možnost zobrazení všech jejích záznamů podle primárních klíčů.

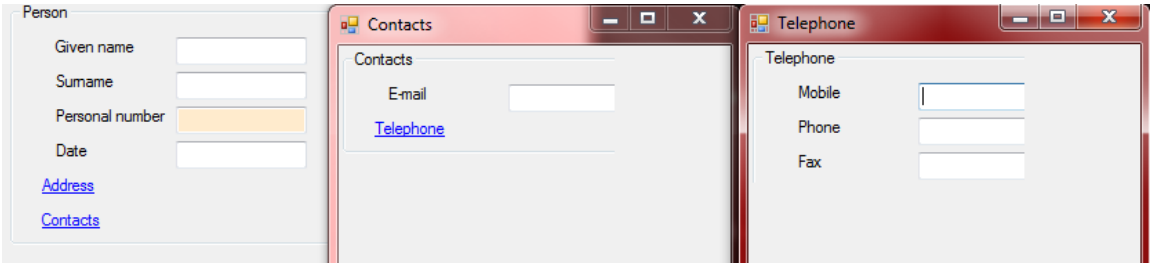
Výsledek: Načtení proběhlo bez potíží. Uživatel musí sám kontrolovat jestli má v aplikace správná metadata, jinak může dojít k nekorektnímu zobrazení dat (například nezobrazení některých položek).

Test 17: Změna způsobu zobrazování formulářů

Postup:
Pomocí menu View změnit způsob zobrazování na New Window View.

Očekávaný výsledek:
Překreslení formulářů do zobrazení, které do jednoho okna vypisuje prvky jedné úrovně a ty, které mají podstruktury zobrazí jako hypertextový odkaz, který otevře nové okno, zase s jednou úrovní prvků a hypertextovými odkazy.

Výsledek:
Formuláře se zobrazují korektně ve všech úrovních.



Obrázek C.10: GUI Test 16 - zobrazení ve více oknech

Tento zobrazovací mód funguje pouze pro zobrazování. Nelze použít pro úpravy dat ani metadat.